

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BOARD OF PATENT APPEALS AND INTERFERENCES

In re application of:
Oleg KOUTYRINE et al.

For: SYSTEM AND METHOD FOR
SELECTIVE LOCAL OBJECT
RETRIEVAL

Filed: October 23, 2003

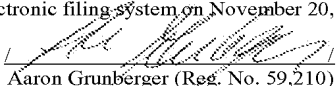
Serial No.: 10/693,178

Examiner: John Q. Chavis

Art Unit: 2193

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

I hereby certify that this correspondence is being electronically transmitted to the United States Patent and Trademark Office via the Office electronic filing system on November 20, 2007.

Signature: 
Aaron Grunberger (Reg. No. 59,210)

APPEAL BRIEF PURSUANT TO 37 C.F.R. § 41.37

SIR:

On September 20, 2007, Appellants submitted a Notice of Appeal from the last decision of the Examiner contained in the Final Office Action dated April 9, 2007 in the above-identified patent application.

In accordance with 37 C.F.R. § 41.37, this brief is submitted in support of the appeal of the final rejection of claims 1 to 16. For at least the reasons set forth below, the final rejection of claims 1 to 16 should be reversed.

1. REAL PARTY IN INTEREST

The real party in interest in the present appeal is SAP Aktiengesellschaft (SAP) of Walldorf, Germany, which is the assignee of the entire right, title and interest in the present application.

2. RELATED APPEALS AND INTERFERENCES

There are no other prior or pending appeals, interferences or judicial proceedings known by the undersigned, or believed by the undersigned to be known to Appellants or the assignee, SAP, “which may be related to, directly affect or be directly affected by or have a bearing on the Board’s decision in the pending appeal.”

3. STATUS OF CLAIMS

Claims 17 to 21 have been withdrawn from consideration.

Claims 1 to 16 stand rejected under 35 U.S.C. § 102(e) as anticipated by U.S. Patent No. 6,922,685 (“Greene et al.”).

Appellants appeal from the final rejection of claims 1 to 16.

A copy of the appeal claims, *i.e.*, claims 1 to 16, is attached hereto in the Claims Appendix.

4. STATUS OF AMENDMENTS

In response to the Final Office Action dated April 9, 2007, Appellants submitted an Amendment dated June 11, 2007. The Advisory Action dated July 20, 2007 stated that the proposed amendments included in the Amendment dated June 11, 2007 would not be entered. Accordingly, it is believed that the amendments were not entered and claims 1 to 17 remain pending in the present application.

5. SUMMARY OF THE CLAIMED SUBJECT MATTER

The presently claimed subject matter of independent claims 1, 5, 6, 7, 9, 13, 14, and 16 relate to a computer system or computer-implemented method for selectively retrieving runtime objects in an application development environment. *Specification*, paragraph 1, and Figure 1. In particular, the claims relate to a system that allow for local retrieval of runtime objects, wherever available and valid, even in an object development environment in which developers often cause generation of the same runtime objects according to different generation states. *Specification*, paragraphs 1 to 6.

Independent claims 1, 5, 9, and 16 provide for storing, or a first memory unit or a server database configured to store, a plurality of server runtime objects, and storing, or a second memory unit or a local database configured to store, a plurality of local runtime objects. *Specification*, paragraphs 80 and 81, and Figure 17. Claims 1 and 9 provide that each local runtime object includes a generation setting associated with generation of the

respective local runtime object. *Specification*, paragraph 89, and Figure 22. Claim 16 provides that each local runtime object includes a respective identification of one of a plurality of generation settings, the identified generation setting being associated with the generation of the respective local runtime object. *Specification*, paragraph 89. Claims 1, 9, and 16 provide for responding, or a processor configured to respond, to a request for a requested runtime object by retrieving a valid copy of the requested runtime object from the plurality of local runtime objects if therein, and to otherwise retrieve the valid copy of the requested runtime object from the plurality of server runtime objects. *Specification*, paragraph 85, and Figure 20.

Independent claim 5 provides a generator component configured to include first and second generator components, each provided in communication with each other and the server and local databases. *Specification*, paragraph 94, and Figure 25. The generator component is configured to invalidate and validate server and local runtime objects, to retrieve a requested one of the server and local runtime objects, and to regenerate the requested runtime object conditional upon the retrieved runtime object's invalidity. *Specification* paragraphs 85 and 95. The first generator is configured to perform a first portion of the generator component's tasks and the second generator is configured to perform a second portion of the generator component's tasks. *Specification*, paragraph 94.

Independent claims 6 and 13 provides for storing, or a local database to store, a plurality of pointers and a plurality of local runtime objects, at least one local runtime object from the plurality of local runtime objects including a content, a state, and an original checksum attribute. *Specification*, paragraph 93, and Figures 23 and 24. The original checksum attribute is configured to represent a combination of the content and the state of the local runtime object with which the original checksum attribute is associated. *Specification*, paragraph 92. At least one pointer is configured to identify, from the plurality of local runtime objects, a local runtime object including an original checksum attribute. *Specification*, paragraphs 92 and 93. The at least one pointer is configured to include a copy of the original checksum attribute associated with the local runtime object that the at least one pointer is configured to identify. *Specification*, paragraph 93, and Figure 24. Claims 6 and 13 further provide for responding, or a generator component to respond, to a request for a local runtime object by: calculating a new checksum attribute associated with the requested local runtime object, comparing the requested local runtime object's new checksum attribute to its pointer's copy of the original checksum attribute, invalidating the requested local runtime object when the new checksum attribute and the copy of the original checksum

attribute do not match, and retrieving the requested local runtime object if the requested local runtime object remains valid. *Specification*, paragraph 93.

Independent claims 7 and 14 provide for storing, or a server database configured to store, a plurality of server runtime objects, and storing, or a local database configured to store, a plurality of local runtime objects, each local runtime object from the plurality of local runtime objects configured to correspond to one server runtime object from the plurality of server runtime objects. *Specification*, paragraphs 80 to 82, and Figure 17. Claims 7 and 14 further provide for responding, or a generator component that is responsive, to a request for a requested runtime object by being configured to: determine if the plurality of server runtime objects includes a valid copy of the requested runtime object; if it is determined that the plurality of server runtime objects includes the copy of the requested runtime object, determine if the plurality of local runtime objects includes a runtime object that corresponds to the valid copy of the requested runtime object; and retrieve the requested runtime object from the plurality of local runtime objects if it is determined that the plurality of local runtime objects includes the runtime object that corresponds to the valid copy of the requested runtime object, and retrieve the valid copy of the requested runtime object from the plurality of server runtime objects otherwise, if therein. *Specification*, paragraph 85, and Figure 20.

The appealed claims include no means-plus-function language and no step-plus-function claims, so that 41.37(c)(1)(v) is satisfied as to its specific requirements for such claims, since none are present here.

6. GROUND OF REJECTIONS TO BE REVIEWED ON APPEAL

Whether claims 1 to 16, which stand rejected under 35 U.S.C. § 102(e), are anticipated by Greene et al.

7. ARGUMENTS

Rejection of Claims 1 to 16 Under 35 U.S.C. § 102(e)

Claims 1 to 16 stand rejected under 35 U.S.C. § 102(e) as anticipated by Greene et al. It is respectfully submitted that Greene et al. do not anticipate any of claims 1 to 16 for at least the following reasons.

It is “well settled that the burden of establishing a *prima facie* case of anticipation resides with the [United States] Patent and Trademark Office.” *Ex parte Skinner*, 2 U.S.P.Q.2d 1788, 1788 to 1789 (Bd. Pat. App. & Inter. 1986). To reject a claim under 35

U.S.C. § 102(e), the Office must demonstrate that each and every claim feature is identically described or contained in a single prior art reference. (*See Scripps Clinic & Research Foundation v. Genentech, Inc.*, 18 U.S.P.Q.2d 1001, 1010 (Fed. Cir. 1991)). As explained herein, the Final Office Action does not meet this standard as to all of the features of the claims.

A. Claims 1, 3, 4, 9, 11, 12, and 16

Claim 1 relates to a computer system for selectively retrieving runtime objects in an application development environment. Claim 1 provides for:

. . . a second memory unit storing a plurality of local runtime objects, each local runtime object including a generation setting associated with generation of the respective local runtime object . . .

Claim 9 relates to a computer-implemented method for selectively retrieving runtime objects in an application development environment and includes subject matter analogous to that of claim 1. Claim 16 relates to a computer system for selectively retrieving runtime objects in an application development environment and includes subject matter analogous to that of claim 1.

The Examiner refers to a “self-healing” function discussed in column 60, and to column 61, lines 62 to 66, and column 62, lines 21 to 30 of Greene et al. as allegedly disclosing these features of claims 1, 9, and 16. The referenced sections discuss copies of shared data objects and entity instance objects used for accessing the shared data objects, but Greene et al. do not disclose, or even suggest, that any of these objects includes a generation setting of any kind, and certainly not one that is associated with the object’s generation.

The basis of the Examiner’s rejection of these claims is an unreasonably broad interpretation of the term “generation setting.” In this regard, the Final Office Action interprets “generation setting”¹ as “anything that can help identify components for debugging.” Final Office Action, page 4. However, during patent examination, the claims are given the broadest reasonable interpretation consistent with the specification. *See In re Morris*, 127 F.3d 1048, 44 U.S.P.Q.2d 1023 (Fed Cir. 1997). The words of the claim must be given their plain meaning unless applicant has provided a clear definition in the specification. *M.P.E.P.* § 2111.01. The Final Office Action’s interpretation is overly broad in view of both the specification and the plain meaning of the term “generation setting” for the following reasons.

¹ The Office Action actually interprets the term “generation information,” but it appears that the Office Action intended to interpret the term “generation setting” since the latter is the term used in the claims.

The Final Office Action incorrectly asserts that the present application's specification supports the Final Office Action's interpretation. The Final Office Action states that "[t]he Applicant indicates that a generation setting can be debug information, etc, and later indicate that his system indicates a generation timestamp." Final Office Action, page 4. These characterizations of the specification are incorrect. The specification indicates that a generation setting is a setting according to which generation is performed. *See* specification, e.g., at paragraph 1.

As for the Final Office Action's reference to debug information, the specification indicates that the generation settings can include a setting that indicates how an object is to be generated with respect to the feature of including debug information. For example, generation settings may include the setting of "Include additional Debug Code," indicating whether an object is to be generated with the attribute of including additional debug code. *See* specification, e.g., at paragraph 86. In response to the preceding argument regarding debug information, the Examiner, in the Advisory Action, asserts that "the applicant also indicates that his generation settings have nothing to do with 'debugging information'; however, see the applicant's specifications section 0001." Advisory Action, page 2. The Examiner misrepresents and does not address Appellants' actual argument. Appellants did not state that a generation setting "has nothing to with" debugging information, (though this may be the case for some generation settings). Rather, Appellants assert that debug information itself is not a generation setting. Instead, at most, a generation setting can be a setting regarding debug information.

As for the Final Office Action's reference to inclusion of a generation timestamp, an object's inclusion of a generation timestamp does not shed any light on the meaning of the term "generation setting."

Further, the correct interpretation of "generation setting" as a setting according to which generation is performed is in line with the plain meaning of "setting" in the field of computers. A computer setting refers to one or more variables which are set and the states of which indicate how the computer is to behave with respect to aspects to which the variables correspond. For example, a resolution setting indicates the resolution according to which a computer is to generate a display.

Therefore, the Examiner's interpretation of the term "generation setting" and, therefore, the basis of the rejection of claims 1, 9, and 16, is clearly erroneous. Therefore, the Examiner has never addressed the features actually recited in claims 1, 9, and 16 and has not set forth a *prima facie* case of anticipation. Indeed, it is respectfully submitted that

Greene et al. do not identically disclose, or even suggest, each feature recited in any of claims 1, 9, and 16, so that Greene et al. do not anticipate any of claims 1 (and its dependent claims, e.g., claims 3 and 4), 9 (and its dependent claims, e.g., claims 11 and 12), and 16.

Reversal of this rejection with respect to claims 1, 3, 4, 6, 11, 12, and 16 is therefore respectfully requested.

B. Claims 2 and 10

Claim 2 depends from claim 1 and therefore includes all of the features of claim 1. Claim 10 depends from claim 9 and therefore includes all of the features of claim 9. Accordingly, Greene et al. do not anticipate these claims for at least the same reasons set forth above in support of the patentability of claims 1 and 9, respectively.

Furthermore, claim 2 provides that the processor is configured to invalidate a local runtime object when the local runtime object's generation setting does not match a current generation setting. Claim 10 includes subject matter analogous to that of claim 2.

The Examiner incorrectly asserts that the "self-healing" function of Greene et al. discloses a processor that is configured to invalidate an object when the object's generation setting does not match a current generation setting. The "self-healing" function of Greene et al. is performed to update registrars including stale references to old services that are no longer available. During the function, if it is determined that the registrar includes such a stale reference, the reference is removed. *See* Greene et al., column 57, line 66 to column 58, line 56 and column 59, line 58 to column 60, line 19. The "self-healing" function does not disclose, or even suggest, a matching of a current generation setting to one associated with an object's generation, even according to the Final Office Action's incorrect interpretation of the term "generation setting," and certainly according to the correct interpretation of this term.

For this additional reason, Greene et al. do not disclose, or even suggest, all of the features recited in either of claims 2 and 10, and do not anticipate either of claims 2 and 10.

In view of all of the foregoing, reversal of this rejection with respect to claims 2 and 10 is respectfully requested.

C. Claim 5

Claim 5 recites, in part, “the generator component configured to . . . retrieve a requested [runtime object], and to regenerate the requested runtime object conditional upon the retrieved runtime object’s invalidity.”

The Examiner has not, in any of the Office Actions, addressed these features. Indeed, no aspect of Greene et al. identically discloses, or even suggests, regeneration of a requested runtime object.

In the Advisory Action, the Examiner (for the first time) asserts that regeneration is an inherent feature of synchronization. As an initial matter, the synchronization of Greene et al. on which the Examiner intends to rely as assertedly necessarily including a regeneration of a requested runtime object is not apparent. Further, to rely on inherency, the Examiner must provide a “basis in fact and/or technical reasoning to reasonably support the determination that the allegedly inherent characteristics *necessarily* flows from the teachings of the applied art.” (See M.P.E.P. § 2112; emphasis in original; and see *Ex parte Levy*, 17 U.S.P.Q.2d 1461, 1464 (Bd. Pat. App. & Int’f. 1990)). Thus, the M.P.E.P. and the case law make clear that simply because a certain result or characteristic may occur in the prior art does not establish the inherency of that result or characteristic. The Examiner has not provided any basis in fact or technical reasoning to support the assertion that a synchronization necessarily includes a regeneration. Indeed, it is respectfully submitted that a synchronization does not necessarily include a regeneration.

Further, even if regeneration is deemed an inherent feature of a synchronization, the Examiner has not at all addressed the feature of retrieving a runtime object and regenerating a requested runtime object “conditional upon the *retrieved* runtime object’s invalidity,” so that the Examiner has not set forth a *prima facie* case of anticipation. Indeed, it is apparent that the Examiner has never considered these features. It is respectfully submitted that no aspect of Greene et al. identically discloses, or even suggests, these features.

In view of all of the foregoing, reversal of this rejection with respect to claims 2 and 10 is respectfully requested.

D. Claims 6 and 13

Each of independent claims 6 and 13 recites, in part, “at least one local runtime object from the plurality of local runtime objects including a content, a state, and an original checksum attribute.”

No aspect of Greene et al. identically discloses, or even suggests, a local runtime object which includes an original checksum attribute. In the Advisory Action, the Examiner asserts (for the first time) that “Greene provides various information that must inherently be checked for security purposes in order to synchronize copies of data between systems.” Advisory Action, page 2. Even if checking of data for security purposes in order to synchronize copies of data between systems is inherently provided by Green et al. (which Appellants do not concede), it is not clear how this further discloses checksums. In this regard, it is noted that a checksum is a particular type of error-detection scheme. See, for example, <http://www.webopedia.com/TERM/c/checksum.html>. The Examiner has not provided any basis in fact and/or technical reasoning to reasonably support the determination that a data checking that includes use of a checksum is necessarily provided in the system of Greene et al.

The Examiner further asserts in the Advisory Action (for the first time) that “the primary keys [of Greene et al.] are also considered pointers which are also used to provide for inherent checksums via . . . (filters) [and] Greene’s version number comparisons are also considered to provide checksums.” Advisory Action, page 2. It is clear that the Office misunderstands the term checksum to refer to any kind of error detection. Without addressing whether Greene et al. disclose or inherently include error detection, it is respectfully submitted that Greene et al. do not disclose or inherently provide a checksum.

Thus, the Examiner’s position with respect to claims 6 and 13 lacks any factual basis and demonstrates clear error.

Further, each of independent claims 6 and 13 recites a number of features that pertain to a particular way in which a checksum attribute is used. As indicated above, Greene et al. do not identically disclose, or even suggest, a checksum attribute, and therefore certainly do not identically disclose, or even suggest, the particular manner in which to use a checksum attribute as provided for in each of claims 6 and 13.

Other than to generally (and incorrectly) assert that Greene et al. disclose a checksum, the Examiner has not at all addressed the particular features of claims 6 and 13 regarding the use of the checksum. For example, the Examiner has never addressed the features of (1) an original checksum attribute which is configured to represent a combination

of the content and the state of a runtime object, (2) a pointer configured to identify a runtime object including a checksum attribute and to include a copy of the checksum attribute, or (3) a generator component that, in response to a request for a runtime object, calculates a new checksum attribute, compares the new checksum attribute to its pointer's copy of the original checksum attribute, and invalidates the requested runtime object when the checksum attributes do not match. Since the Examiner has never addressed these features, the Examiner has not set forth a *prima facie* case of anticipation, and it is apparent that the Examiner has never considered these features. Indeed, it is respectfully submitted that Greene et al. do not identically disclose, or even suggest, these features, and therefore do not anticipate either of claims 6 and 13.

In view of all of the foregoing, reversal of this rejection with respect to claims 6 and 13 is respectfully requested.

E. Claims 7, 8, 14, and 15

Independent claim 7 recites, in part, “a generator component . . . configured to . . . if it is determined that the plurality of server runtime objects includes the copy of the requested runtime object, determine if the plurality of local runtime objects includes a runtime object that corresponds to the valid copy of the requested runtime object.” Claim 14 includes similar subject matter.

The Examiner has not at all addressed these features, and therefore has not set forth a *prima facie* case of anticipation. Indeed, no aspect of Greene et al. identically discloses, or even suggests, these features.

Thus, Greene et al. do not anticipate either of claims 7 and 14 (or their dependent claims 8 and 15, respectively).

Reversal of this rejection with respect to claims 7, 8, 14, and 15, is therefore respectfully requested.

8. CLAIMS APPENDIX

A “Claims Appendix” is attached hereto and appears on the page labeled “Claims Appendix.”

9. EVIDENCE APPENDIX

No evidence has been submitted pursuant to 37 C.F.R. §§ 1.130, 1.131 or 1.132. No other evidence has been entered by the Examiner and relied upon by Appellants in the appeal. An “Evidence Appendix” is attached hereto.

10. RELATED PROCEEDINGS APPENDIX

As indicated above in Section 2, above, “[t]here are no other prior or pending appeals, interferences or judicial proceedings known by the undersigned, or believed by the undersigned to be known to Appellants or the assignee, SAP, ‘which may be related to, directly affect or be directly affected by or have a bearing on the Board’s decision in the pending appeal.’” As such, there are no “decisions rendered by a court or the Board in any proceeding identified pursuant to [37 C.F.R. § 41.37(c)(1)(ii)]” to be submitted. A “Related Proceedings Appendix” is nevertheless attached hereto.

11. CONCLUSION

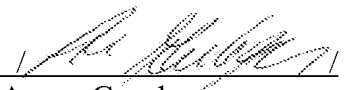
For at least the reasons indicated above, Appellants respectfully submit that the art of record does not disclose or suggest the subject matter as recited in the claims of the above-identified application. Accordingly, it is respectfully submitted that the subject matter recited in the claims of the present application is new, non-obvious and useful.

In view of all of the foregoing, reversal of all of the rejections set forth in the Final Office Action is therefore respectfully requested.

Respectfully submitted,

Dated: November 20, 2007

By:


Aaron Grunberger
Reg. No. 59,210

KENYON & KENYON LLP
One Broadway
New York, New York 10004
(212) 425-7200

CUSTOMER NO. 26646

CLAIMS APPENDIX

1. A computer system for selectively retrieving runtime objects in an application development environment, comprising:
 - a first memory unit storing a plurality of server runtime objects; and
 - a second memory unit storing a plurality of local runtime objects, each local runtime object including a generation setting associated with generation of the respective local runtime object; and
 - a processor responsive to a request for a requested runtime object by being configured to retrieve a valid copy of the requested runtime object from the plurality of local runtime objects if therein, and to otherwise retrieve the valid copy of the requested runtime object from the plurality of server runtime objects if therein.
2. The system of claim 1, wherein the processor is configured to invalidate a local runtime object when the local runtime object's generation setting does not match a current generation setting.
3. The system of claim 1, wherein the processor is configured to retrieve the local runtime object by returning a data element indicating the requested local runtime object's validity.
4. The system of claim 1, further comprising a local database including a first data structure and a second data structure, the second data structure configured to store a plurality of pointers, at least one pointer configured to identify a local runtime object from the plurality of local runtime objects, the first data structure configured to store a plurality of commands, the commands configured to manipulate the second data structure.

5. A computer system for selectively retrieving runtime objects in an application development environment, comprising:

- a server database configured to store a plurality of server runtime objects;
- a local database configured to store a plurality of local runtime objects; and
- a generator component configured to include a first generator and a second generator, each provided in communication with each other and the server and local databases,

- the generator component configured to invalidate and validate server and local runtime objects, to retrieve a requested one of the server and local runtime objects, and to regenerate the requested runtime object conditional upon the retrieved runtime object's invalidity,

- the first generator configured to perform a first portion of the generator component's tasks, the second generator configured to perform a second portion of the generator component's tasks.

6. A computer system for retrieving stored runtime objects in an application development environment, comprising:

- a local database to store a plurality of pointers and a plurality of local runtime objects, at least one local runtime object from the plurality of local runtime objects including a content, a state, and an original checksum attribute,

- the original checksum attribute configured to represent a combination of the content and the state of the local runtime object with which the original checksum attribute is associated,

- at least one pointer configured to identify, from the plurality of local runtime objects, a local runtime object including an original checksum attribute,

- the at least one pointer configured to include a copy of the original checksum attribute associated with the local runtime object that the at least one pointer is configured to identify; and

- a generator component to, in response to a request for a local runtime object:

- calculate a new checksum attribute associated with the requested local runtime object,

- compare the requested local runtime object's new checksum attribute to its pointer's copy of the original checksum attribute,

- invalidate the requested local runtime object when the new checksum attribute and the copy of the original checksum attribute do not match, and

retrieve the requested local runtime object if the requested local runtime object remains valid.

7. A computer system for selectively retrieving runtime objects in an application development environment, comprising:

- a server database configured to store a plurality of server runtime objects;

- a local database configured to store a plurality of local runtime objects, each local runtime object from the plurality of local runtime objects configured to correspond to one server runtime object from the plurality of server runtime objects; and

- a generator component responsive to a request for a requested runtime object by being configured to:

 - determine if the plurality of server runtime objects includes a valid copy of the requested runtime object;

 - if it is determined that the plurality of server runtime objects includes the copy of the requested runtime object, determine if the plurality of local runtime objects includes a runtime object that corresponds to the valid copy of the requested runtime object; and

 - retrieve the requested runtime object from the plurality of local runtime objects if it is determined that the plurality of local runtime objects includes the runtime object that corresponds to the valid copy of the requested runtime object, and to retrieve the valid copy of the requested runtime object from the plurality of server runtime objects otherwise, if therein.

8. The system of claim 7,

wherein at least one local runtime object from the plurality of local runtime objects and at least one server runtime object from the plurality of server runtime objects contain a content attributes and a state attributes, and

wherein the each local runtime object corresponds to the one server runtime object when the each local runtime object's content attributes and state attributes match the one server runtime object's content attributes and state attributes.

9. A computer-implemented method for selectively retrieving runtime objects in an application development environment, comprising:

storing a plurality of server runtime objects;

storing a plurality of local runtime objects, each local runtime object including a generation setting associated with generation of the respective local runtime object; and

responding to a request for a requested runtime object by retrieving a valid copy of the requested runtime object from the plurality of local runtime objects if therein, and otherwise by retrieving the valid copy of the requested runtime object from the plurality of server runtime objects if therein.

10. The method of claim 9, further comprising:

invalidating a local runtime object when the local runtime object's generation setting does not match a current generation setting.

11. The method of claim 9, wherein the local runtime object is retrieved by returning a data element indicating the requested local runtime object's validity.

12. The method of claim 9, further comprising:

storing a plurality of pointers, at least one pointer identifying a local runtime object from the plurality of local runtime objects; and

storing a plurality of commands, the commands manipulating the plurality of pointers.

13. A computer-implemented method for retrieving runtime objects in an application development environment, comprising:

storing a plurality of pointers;

storing a plurality of local runtime objects, at least one local runtime object from the plurality of local runtime objects including a content, a state, and an original checksum attribute,

the original checksum attribute representing a combination of the content and the state of the local runtime object with which the original checksum attribute is associated,

at least one pointer identifying, from the plurality of local runtime objects, a local runtime object including an original checksum attribute,

the at least one pointer including a copy of the original checksum attribute associated with the local runtime object that the at least one pointer identifies; and

responding to a request for a local runtime object by:
calculating a new checksum attribute associated with the requested local runtime object,
comparing the requested local runtime object's new checksum attribute to its pointer's copy of the original checksum attribute,
invalidating the requested local runtime object when the new checksum attribute and the copy of the original checksum attribute do not match, and
retrieving the requested local runtime object if the requested local runtime object remains valid.

14. A computer-implemented method for selectively retrieving runtime objects in an application development environment, comprising:

storing a plurality of server runtime objects;
storing a plurality of local runtime objects, each local runtime object from the plurality of local runtime objects capable of corresponding to one server runtime object from the plurality of server runtime objects; and

responding to a request for a requested runtime object by:

determining if the plurality of server runtime objects includes a valid copy of the requested runtime object;

if it is determined that the plurality of server runtime objects includes the copy of the requested runtime object, determining if the plurality of local runtime objects includes a runtime object that corresponds to the valid copy of the requested runtime object; and

retrieving the requested runtime object from the plurality of local runtime objects if it is determined that the plurality of local runtime objects includes the runtime object that corresponds to the valid copy of the requested runtime object, and by retrieving the valid copy of the requested runtime object from the plurality of server runtime objects otherwise, if therein.

15. The method of claim 14,
wherein at least one local runtime object from the plurality of local runtime objects and at least one server runtime object from the plurality of server runtime objects contain a content attributes and a state attributes, and

wherein the each local runtime object corresponds to the one server runtime object when the each local runtime object's content attributes and state attributes match the one server runtime object's content attributes and state attributes.

16. A computer system for selectively retrieving runtime objects in an application development environment, comprising:

a first memory unit storing a plurality of server runtime objects; and

a second memory unit storing a plurality of local runtime objects generated based on corresponding server runtime objects, each local runtime object including a respective identification of one of a plurality of generation settings, the identified generation setting being associated with the generation of the respective local runtime object; and
a processor configured to respond to a request for a requested runtime object by retrieving a valid copy of the requested runtime object from the plurality of local runtime objects if therein, and to otherwise retrieve the valid copy of the requested runtime object from the plurality of server runtime objects if therein.

EVIDENCE APPENDIX

No evidence has been submitted pursuant to 37 C.F.R. §§1.130, 1.131, or 1.132. No other evidence has been entered by the Examiner and relied upon by Appellants in the appeal.

RELATED PROCEEDINGS APPENDIX

As indicated above in Section 2 of this Appeal Brief, “[t]here are no other prior or pending appeals, interferences or judicial proceedings known by the undersigned, or believed by the undersigned to be known to Appellants or the assignee, SAP, ‘which may be related to, directly affect or be directly affected by or have a bearing on the Board’s decision in the pending appeal.’” As such, there no “decisions rendered by a court or the Board in any proceeding identified pursuant to [37 C.F.R. § 41.37(c)(1)(ii)]” to be submitted.